

LINUX EXPERT

Optimise your site for search engines

Your website is only ever as successful as its most recent hit. **Dan Goscomb** explains how search engine optimisation can ensure that your site gets all the attention it deserves

Your website may be the best of its kind on the internet, with the most accurate text, incredibly artistic images and the fastest-loading pages. But if the main internet search engines don't rate it highly, the millions of potential visitors who could appreciate it will never even find it, let alone revisit. To make sure that your site becomes and stays popular, you need to attract and maintain the attention of search engines such as Google. Furthermore, you need these services to list your site at the top of the results, so the people using them will click through to your site before trying a competitor. We'll show you how to start optimising your site for search engines, which should help increase your traffic and potentially your income.

Making a site work better with search engines is called search engine optimisation (SEO). Google and other sites want to display links to the most relevant sites at the top of their results lists, and there are ways to increase your chances of appearing there, or at least on the first page of results. Some of these ways involve tricking the search engine, but these aren't worth trying; the staff who run the search systems become aware of these tricks quickly and will penalise your site when they detect you are trying to pull a fast one. The best approach is to provide genuinely high-quality content and to make the site easy for users. The search engine programmers will rate such sites higher, you will attract more links from other sites, and that in turn will impress search engines such as Google even more.

The SEO technique we'll cover here involves the web addresses of pages on your site. Often you'll find sites that display pages by pulling information from a database. These sites will often show URLs such as `www.yoursite.com/article.php?id=23`, which is ugly and doesn't tell us much about what we'll find if we click on the link. A URL such as `www.yoursite.com/articles/23/search_engine_optimisation.html` makes it obvious that clicking on the link will display an article about SEO. Not only does this make it more attractive to users, but search engines often prefer this approach, too. We're going to show you how to change your site from one

that displays meaningless URLs to one that shows useful, search engine-friendly links.

FRIENDLY URLs

When a web server receives a request for a page it looks for an appropriate file, or runs a script to generate the page, before sending the data to the visitor's web browser. Attempting to access `www.yoursite.com/article.php?id=23` will cause the server to run the `article.php` script that will, in this case, display an article stored in a database. This article's id code will be 23.

It would be better if the web address was more readable by people, though. We need to be able to allow people to ask for `www.yoursite.com/articles/23/search_engine_optimisation.html` and have the web server read this, convert it into a request that the server can handle (such as the previous `article.php?id=23` URL) and then show the resulting page. The Apache web server has a module that can do just that. It is called `mod_rewrite`, which is very flexible. It can be used for a number of tasks, from making URLs search-engine-friendly to stopping third-party websites stealing your bandwidth by linking directly to your images.

To use the `mod_rewrite` features, you must either have root access to your system, so you can edit the Apache `httpd.conf` file, or access to a machine where you have permissions to create `mod_rewrite` rules in `.htaccess` files. Most hosting providers allow this but, if you are unsure, email the support department to check. The commands for each method are the same. There are slight differences in using the `httpd.conf` file directly, in that you have to put the commands within a certain `VirtualHost` section and also restart the server after you have made any changes.

`Mod_rewrite` is included in the standard Apache installation, so if you have Apache running then you should have `mod_rewrite`, too. If you compiled Apache yourself and chose not to install the rewrite modules, you will need to recompile Apache with the appropriate flags in the configure script. With Apache 1.3, these would be `--enable-module=rewrite` and `--enable-`

FAQ

Q I have seen a lot of articles and internet forums about SEO. What is it, and is it worth learning?

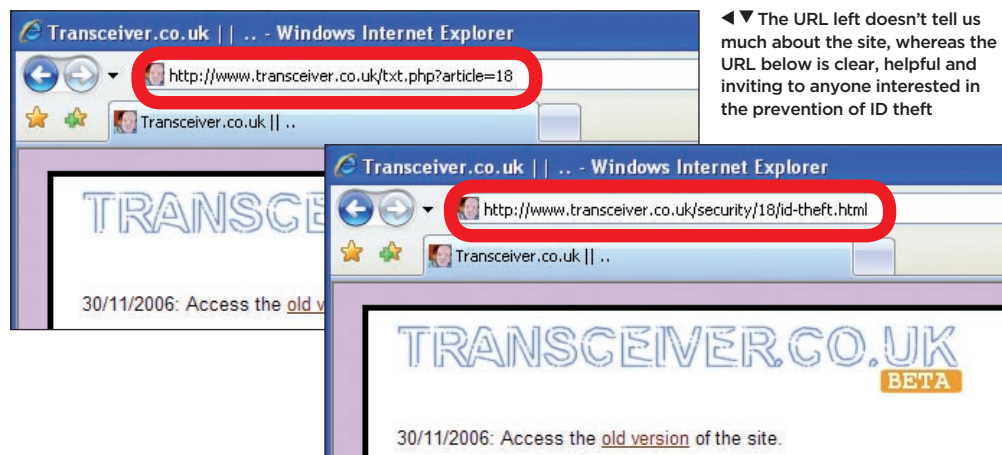
A SEO stands for search engine optimisation. This involves doing things to your website so it appears near the top of search engines' results. Techniques include making your site easier for search engines to index, convincing other sites to link to your pages and ensuring that the content you provide is written in such a way that search engines can understand your site accurately.

Generally speaking, if you make your site easy to use for everyone, from PC users to those browsing with mobile phones, and don't try to cheat by trading links with unrelated sites or creating lots of fake sites, search engines such as Google will place your site high on the list of relevant search results.

Dan Goscomb

Linux administrator,
programmer and ISP

linux@computershopper.co.uk





shared=rewrite. With Apache 2, you would use --enable-rewrite.

UNDERSTANDING THE RULES

Mod_rewrite uses a rule-based language to rewrite URLs on the fly. When a request is made to the web server, mod_rewrite intercepts this and processes the URL using one or more user-defined rules to see if a match is found. If one is found, the requested URL is replaced with the one specified in the rule. This is then passed on to the server to process as usual.

You can allow URL rewriting on one or more virtual hosts. To turn on URL rewriting for a specific site on your server, you should insert the following line into its set of configuration directives. Add it to the general set of rules if you want to enable it for all servers, or if you don't use virtual servers:

RewriteEngine On

If you use .htaccess files, you may need an extra line if your server does not have the FollowSymLinks option set in the main configuration:

```
Options +FollowSymLinks
RewriteEngine On
```

It is important that these lines are listed in this order within your .htaccess files, as the 'Options +FollowSymLinks' line affects how mod_rewrite works. Unless these lines are present, any of the following mod_rewrite rules will not work. Once the engine is turned on, you can specify one or more rules for processing your URLs.

Rewrite rules are based on regular expressions (see the PDF of *Shopper* May 2007's *Linux Expert* on the cover disc) so you can extract pieces of information from the initial requested URL and use them to form part of the real URL that the server will understand how to process. In effect, you're creating a dynamic mapping of fake URLs that map gracefully on to real files on the server.

Rules are processed by the mod_rewrite engine in the order they are read (from top to bottom). If a match is not found on a rule, the next rule is processed immediately. If a match is found, any rule conditions present are checked. If they all match, the URL is rewritten according to the rule. The system then moves on to the next rule, unless the rule it has just processed was specified as being a Last rule, or if an external redirect to another site has been used.

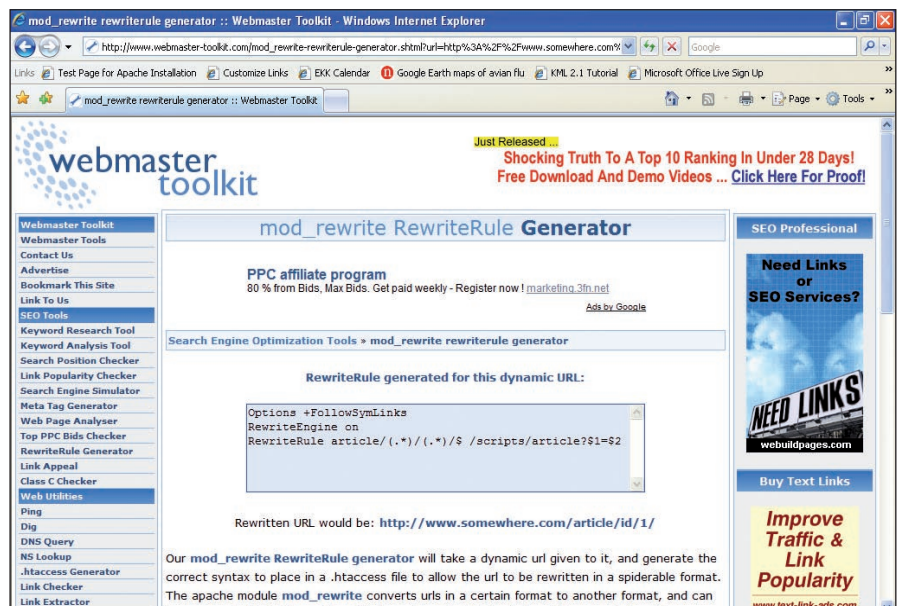
Rules are formatted as follows:

```
RewriteRule <Match Pattern> <Rewritten URL>
[<Flags>]
```

If you want to get an idea how to create rules, try playing with the mod_rewrite RewriteRule Generator at www.webmaster-toolkit.com. This allows you to create the correct code to paste into .htaccess files quickly, although you should read on to see how to do this manually to avoid mistakes and to create powerful, flexible rules.

THE MATCH PATTERN

This is a regular expression that the rewrite system compares against the original URL, attempting to find any matches. You may want to use parts of the match as variables in the rewritten URL. If so, these should be enclosed in brackets. Here is an example of a suitable regular expression:



▲ The mod_rewrite RewriteRule Generator at Webmaster Toolkit (www.webmaster-toolkit.com) lets you create the correct code to paste into .htaccess files quickly

```
^/articles/([0-9]+)/.*$
```

This pattern will match a URL starting with /articles/, followed by a number of one or more characters in length, followed by a / followed by anything. As the number match has been enclosed in brackets, and this is the first time in the expression that this has been done, the value of the match will be placed in the variable \$1 for use later in the <Rewritten URL> statement. If you use a second match in brackets the variable will be \$2, the third will be \$3 and so on.

THE REWRITTEN URL

This is the text that forms the final URL, which the server understands. Following the example in the introduction, it should be:

```
/article.php?id=$1
```

The variable \$1 contains a value that was specified when the user typed in the original URL. If this contained a number after the /articles/ part, this number is stored as \$1 and is added to the above rewritten URL. The server might see something like /article.php?id=23.

Should you not wish to rewrite the URL but just perform an action, you can do so by simply using the '-' symbol in the <Rewritten URL> space. This could be useful if, for example, you wanted to return a 'Forbidden' error message to the browser when someone tries to access a page they are not supposed to.

The flags part of the rewrite is optional. The flags tell the rewrite rule how to handle the rewriting, and there are a number of options. A full list of these is available on the mod_rewrite reference page at http://httpd.apache.org/docs/2.0/mod/mod_rewrite.html#rewriterule.

Multiple option flags can be specified in a comma delimited list. The most commonly used options are explained here:

- [NC] Removes case sensitivity from your URL matching rule
- [F] Returns a 403 Forbidden message to the browser
- [L] Causes the rule processing to finish at the point where this flag

is inserted, assuming that it is attached to a rule that matches the original URL. No further rules will be executed

[R=code] An external redirection is returned to the user. Rather than transparently rewriting the URL, this will send the rewritten URL back to the user's browser with a redirect instruction. You can specify the HTTP code to be sent back by using the =code portion. If no code is specified then a 302 (moved temporarily) redirect is returned. You may use any code within the range 300-400

[PT] Sets Apache's internal pointers to use the new URL, allowing the server to process the rewritten URL through other directives such as Alias. If you do not use this and attempt to use some Apache functions later in your configuration to match against the rewritten URL, then these will not work without this flag

[QSA] Appends the query string applied to the script by the user's browser to the rewritten URL, rather than replacing it. This is very useful if you wish to use rewrite rules to add parameters to the query string of a script

YOUR FIRST RULES

From the above descriptions, the complete rule to achieve the outcome of the example in the introduction would be:

```
RewriteRule ^/articles/([0-9]+)/.*$ /article.php?id=$1
```

In the case above, the text following the id number can be anything. In the introduction text, we used the URL www.yoursite.com/articles/23/search_engine_optimisation.html, which contains the title of the article as a fake filename. In fact, the text following /23/ is irrelevant, other than that it provides a clue

about the article's subject – which is good for people but unnecessary for computers. Should you want to make this more foolproof, you could also make your web script (article.php, in this example) accept the filename as an argument and compare this against a field in your database. To achieve this, you would use the following rule:

```
RewriteRule ^/articles/([0-9]+)/(.*)\.html$
/article.php?id=$1&title=$2
```

Given the URL `www.yoursite.com/articles/23/search_engine_optimisation.html`, the parameters assigned would be as follows:

```
$1 = 23
$2 = search_engine_optimisation
```

The rewritten URL would be `www.yoursite.com/article.php?id=23&title=search_engine_optimisation`. We have omitted any options from this rule, as we simply want the URL to be rewritten internally to Apache and to be invisible to the visitor.

URL rewriting is very simple as long as you can write the correct regular expressions to match and extract the variables that you need from the original URL string. The examples that we have looked at so far demonstrate how to make your URLs search-engine-friendly, imitating real files instead of dynamic database queries. You will obviously have to amend the URLs that appear on your website to fit the new schemes that you create using `mod_rewrite`. How you do this depends entirely on the new URL scheme you choose to implement and the way that you create webpages. Just be sure that there is a way to match the data back to each article's database entry.

MULTIPLE REWRITE RULES

To add multiple rewrite rules together, put them one line after the other in sequence. Remember that the changed URL will be sent to the next rule, should it match the previous rule. For that reason, do not always expect every rule to process the original URL.

Make sure you write your rules in the correct order if they are similar; for example, when one is matching `/articles/comments/23` and the other `/articles/23`. In this case, you would want to put the more specific 'comments' rule above the more generic rule looking only for 'articles'. Here is an example rule set for this situation:

```
RewriteRule ^/articles/([0-9]+)/comments/(.*)
\.html$ /article.php?id=$1&title=$2&mode=
comments
```

```
RewriteRule ^/articles/([0-9]+)/(.*)\.html$
/article.php?id=$1&title=$2
```

The first rule searches for the word 'comments' in the URL and rewrites it, setting the 'mode' part of the query string to 'comments'. If we had written these two rules in the opposite order the comments rule would never match the URL, which would always be rewritten by the more generic rule. The standard article page would be returned without the `mode=comments` addition, no matter which of the two URLs was requested.

ADDING CONDITIONS

It is possible to create some advanced rules that use conditions. Conditions will only allow you to execute a rule should a condition on another variable be met. This does not necessarily have to be a regular expression of the path itself.

Imagine that you have your Apache virtual host set up to respond to queries for the domain `http://www.yoursite.com` and also `http://yoursite.com`. Ideally, you would not have both of these domain names showing up in Google or other search engines because they are one site. You can overcome this by rewriting the actual domain name and presenting the user's browser with a permanent redirect message.

Let us assume that you prefer the URL `http://www.yoursite.com`. We only want to rewrite URLs that start with `yoursite.com` and not `www.yoursite.com`. To do this, implement a condition and a rule as follows:

```
RewriteCond %{HTTP_HOST} ^yoursite\.com
[NC]
RewriteRule ^(.*)$ http://www.yoursite.com$1
[R=301]
```

The `RewriteCond` line checks the `HTTP_HOST` environment variable and makes sure it begins with `yoursite.com`. As this is a regular expression, the `'.'` in the domain name must be escaped with a backslash. Environment variables are accessed using the `%{VARIABLE}` syntax. Any standard CGI environment variable can be accessed here and used in rules. See the box below for a list of useful variables at your disposal.

In the above example, the `R` flag in the options forces a 301 (Moved Permanently) redirect code. This tells search engines to index only the destination of the redirect rather than the redirecting page.

Rewrite conditions must always come immediately before the rule that they should apply, and they can be chained using flags in the same way as rewrite rules. If a flag is omitted then `AND` is assumed and all conditions must be met. There are two available flags:

[OR] The opposite of `AND`; if either rule is matched then the following rule(s) will be processed

[NC] Case-insensitivity, the same for rewrite rules

If you implement our rule on your server, replacing `yoursite.com` with your real domain name in the rules, and gracefully restart the server if you are not using `.htaccess` files, you can immediately see the effect of the rule by visiting `http://yoursite.com`. If the rule has been implemented correctly, your browser will be redirected automatically to `http://www.yoursite.com`. Due to the code used in the `RewriteRule`, any URL will be rewritten to the `www.` version of the site. If you were to visit `http://yoursite.com/filename.html`, your browser would be redirected to `http://www.yoursite.com/filename.html`, which means an end to broken links.

MASTER OF YOUR DOMAINS

If you have two domain names pointing to one site and wish search engines to index only one, you could extend the rule like this:

```
RewriteCond %{HTTP_HOST} ^yoursite\.com
[NC,OR]
RewriteCond %{HTTP_HOST} ^yourothersite\
.com [NC,OR]
RewriteCond %{HTTP_HOST} ^www\
.yourothersite\.com [NC]
RewriteRule ^(.*)$ http://www.yoursite.com$1
[R=301]
```

This would rewrite every request for `yoursite.com`, `yourothersite.com` and `www.yourothersite.com` to `www.yoursite.com`. Search engines should then list only this last URL.

We can combine our two rule sets as follows:

```
RewriteCond %{HTTP_HOST} ^yoursite\.com
[OR,NC]
RewriteCond %{HTTP_HOST} ^yourothersite\
.com [OR,NC]
RewriteCond %{HTTP_HOST} ^www\
.yourothersite\.com
RewriteRule ^(.*)$ http://www.yoursite.com$1
[R=301]
RewriteRule ^/articles/([0-9]+)/(.*)\.html$
/article.php?id=$1&title=$2
```

If you implement this set of rules and visit `http://yoursite.com/articles/23/search_engine_optimisation.html`, you will be redirected to `http://www.yoursite.com/articles/23/search_engine_optimisation.html`. Then the second `RewriteRule` redirects this to the appropriate article.php script (`http://www.yoursite.com/article.php?id=23`). If you visit `http://www.yoursite.com/articles/23/search_engine_optimisation.html` directly, only the second rule alone is applied because the `RewriteCond` lines do not apply.

The example above shows just one rewrite rule that is executed on a set of conditions. However, it is possible to make the conditions execute multiple rules by chaining them together, like this:

```
RewriteCond %{HTTP_HOST} ^yoursite\.com
[OR,NC]
RewriteCond %{HTTP_HOST} ^yourothersite\
.com [OR,NC]
RewriteCond %{HTTP_HOST} ^www\.
```

Understanding the environment Useful variables explained

HTTP_HOST	The hostname being used by the client to access the site, such as <code>www.yoursite.com</code>
HTTP_REFERER	When you click on a link to a site, the page hosting that link is the referrer, such as <code>www.anothersite.com/links.html</code>
HTTP_USER_AGENT	Information on the user's browser and operating system versions
QUERY_STRING	The information from the query string
REMOTE_ADDR	The IP address of the connecting machine
REMOTE_HOST	The hostname of the connecting machine
REQUEST_METHOD	The method used to request the page (GET, POST, PUT and so on)
SERVER_ADDR	The server's IP address
SERVER_PORT	The server's port

Put yourself on the map Indexing a website with Google Sitemaps

With websites becoming increasingly more complex, it is important to get them listed in search engines properly. Google's Sitemaps scheme aims to make indexing your site in its index easier. It also lets you set rules that tell Google how and when to update the index.

A sitemap is an XML file describing the structure of your website. You can upload this to Google to tell its website crawlers to index your site in a particular way. By creating the sitemap yourself, rather than relying on Google to scan your site, you can affect the Change Frequency (how often the content on a particular page is likely to change) and Priority levels of pages, setting some as more important than others on the same site.

The Google crawler discovers pages on your site by following links within the site and from other sites. Sitemaps add to this data to allow Google to pick up all URLs listed in the Sitemap. It can learn about those URLs using the associated metadata that you add. It should be noted, however, that just because you include a URL in a sitemap does not mean it will be included in the Google index.

There are several ways to generate sitemaps of your website. Depending on the complexity and features you wish to use, you may well find that one particular method is best for you. It is easy to generate simple sitemaps by hand, although it is even easier to use a sitemap generator such as that available at www.xml-sitemaps.com.

The online version of this tool crawls your site automatically and creates a sitemap file based on the information it receives. You can edit this file to tweak individual page entries to your liking. If your site is more complex you can use the standalone version of this tool, or you may prefer to use the tools released by Google itself. Visit <http://goog-sitemapgen.sourceforge.net>, but be aware that these tools are not for the faint-hearted.

Once you have generated your sitemap, you need to save a copy of this and upload it to your server in the highest-level directory to which you have access. If possible, use the / directory. This

file's URL can then be submitted to Google. To do this, you must log in to Google's webmaster tools area and submit the sitemap file to its servers by entering its URL. Google will then download the sitemap from your server and read the information within it. Should this be successful, the status of your sitemap will be updated within your webmaster tools area.

To gain additional information on your site, you must verify that you are its owner. Do this by following the onscreen instructions. The extra information provided includes crawler errors and indexing statistics, which can be a useful resource when trying to optimise pages.



◀ Once you have generated your sitemap, you need to upload a copy to your server and log in to your Google Webmaster account

```
yourothersite\com
RewriteRule ^/test/(.*)$ /$1 [C]
RewriteRule ^(.*)$ http://www.yoursite.com$1 [R=301]
```

In this example, both RewriteRule statements will be executed only if the conditions above are met. In the first RewriteRule, the directory /test is removed from the path, and the second rule redirects the user to the real website with the re-written URL from the first rule. Should none of the conditions be met, neither of these rules will be executed. This is achieved by using the chaining option flag [C].

Rules must be in a specific order and conditions must come immediately before the rule(s) that they invoke. Should the conditions affect more than one rule, these rules should be chained. Redirects and similar functions should be used at the end of your rule set, as these will cause the rule set to stop processing and further rules once executed.

STOPPING LEECHES

A common problem with running a popular website is that other sites will link directly to your images, displaying them on their pages, giving you no credit for them but using your bandwidth, which costs you money. Mod_rewrite offers an elegant solution to this problem by allowing us to rewrite people's URLs to return a Forbidden code, or to a different image than the one requested. A good example would be a small image that displays your website address and a note that you do not allow external hotlinking of images.

The following rule set, which handles this job,

uses an environment variable called HTTP_REFERER. This variable contains the URL of the site that is hotlinking your image. We can compare this with a list of allowed URLs. We will start off by denying access to images outright:

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://
(www)?yoursite.com/.*$ [NC]
RewriteRule \.(gif|jpe?g|png)$ - [F]
```

This is possible using some advanced matching features – most importantly the ! symbol, which means 'if you don't match the following', as opposed to the default, which is 'do match'. The other more advanced feature introduced is the ? symbol, which is an optional match. As such, the rule set above will check that:

- a) The referrer is NOT empty; AND
- b) The referrer is NOT http://www.yoursite.com/* OR http://yoursite.com/*

If both conditions are met, the RewriteRule statement will look for any filename ending in gif, jpeg, jpg or png and issue a 403 Forbidden message without rewriting the URL (the '-' symbol is used for this function). You can easily add more filenames to the list by separating them with the pipe symbol. It may be wise to add items such as videos, Zip files any other binary files offered for download.

Taking this rule one step further to return a special image to visitors who are not referred from your own site, you can use the following ruleset:

```
RewriteCond %{HTTP_REFERER} !^$
RewriteCond %{HTTP_REFERER} !^http://
(www)?yoursite.com/.*$ [NC]
RewriteRule \.(gif|jpe?g|png)$ /
donotstealmyimages.gif
```

You can use this to advertise your website on sites that attempt to steal your images. This could then encourage visitors to come to your site instead of viewing the pictures on the referring site.

EFFICIENCY IN ACTION

Mod_rewrite is an extremely flexible and powerful tool that can be used for a multitude of web server tasks. By doing this work at the server level, instead of in your own scripts, the computer works far more efficiently and pages should load more quickly. There are plenty of other mod_rewrite examples on the internet, including those at <http://httpd.apache.org/docs/2.0/misc/rewriteguide.html>.

Although powerful, mod_rewrite should be used with care as you can end up sending rewrites into infinite loops or breaking your website by writing invalid rules. Regular expressions can be hard to master at first, but it will benefit you in the long run to learn how to do this, rather than copying examples, so do try to understand the code you are adding to your rules. ☞

Next month

THE MYSQL DATABASE

Using the popular open-source database